



UWS Academic Portal

Technical debt and waste in non-functional requirements documentation

Robiolo, Gabriela; Scott, Ezequiel; Matalonga, Santiago; Felderer, Michael

Published in:
Product-Focused Software Process Improvement

DOI:
[10.1007/978-3-030-35333-9_16](https://doi.org/10.1007/978-3-030-35333-9_16)

Published: 18/11/2019

Document Version
Peer reviewed version

[Link to publication on the UWS Academic Portal](#)

Citation for published version (APA):

Robiolo, G., Scott, E., Matalonga, S., & Felderer, M. (2019). Technical debt and waste in non-functional requirements documentation: an exploratory study. In Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings (pp. 220–235). (Lecture Notes in Computer Science; Vol. 11915). Springer-Verlag. https://doi.org/10.1007/978-3-030-35333-9_16

General rights

Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

If you believe that this document breaches copyright please contact pure@uws.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

“The final authenticated version is available online at https://doi.org/10.1007/978-3-030-35333-9_16.”

Technical Debt and Waste in Non-Functional Requirements Documentation: An Exploratory Study

Gabriela Robiolo¹, Ezequiel Scott², Santiago Matalonga³, and Michael Felderer⁴

¹ LIDTUA (CIC), Facultad de Ingeniera, Universidad Austral, Argentina
`grobiolo@austral.edu.ar`

² Institute of Computer Science, Tartu University, Estonia
`ezequiel.scott@ut.ee`

³ School of Computing, Engineering and Physical Sciences, University of the West of Scotland, United Kingdom
`santiago.matalonga@uws.ac.uk`

⁴ Department of Computer Science, University of Innsbruck, Austria
`michael.felderer@uibk.ac.at`

Abstract Background: It is important to pay attention to non-functional requirements (NFRs). To adequately attend to NFRs, they must be documented - otherwise, developers would not know about their existence. We assume that there exists a positive correlation between the level of importance of an NFR and its documentation. Aims: The goal is to explore the relationship between the level of importance and degree of documentation for NFRs. Method: Based on a subset of data acquired from the most recent NaPiRE (Naming the Pain in Requirements Engineering) survey, we calculate for a standard set of NFR types comprising Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, and Usability how often respondents state they document a specific type of NFR when they also state that this type of NFR is important. In addition, we calculate the occurrence of potential *Technical Debt* and *Waste*. Results: Our analyses based on 398 survey responses indicate that for four NFR types (Maintainability, Reliability, Usability, and Performance), more than 22% of the 398 survey respondents who labelled those NFR types as important stated that they did not document them. We interpret this as an indication that these NFR types have a higher risk of Technical Debt than other NFR types. Regarding Waste, the problem is less frequent, with the exception of Security with 15% of respondents stating that they document requirements which they do not consider important. Conclusions: There is a clear indication that for a fixed set of NFRs lack of documentation of important NFRs occurs often, suggesting a risk of Technical Debt. The potential risk of incurring Waste is also present but to a lesser extent.

Keywords: Non Functional Requirements · Technical Debt · Waste

1 Introduction

Non-functional requirements (NFRs) are of high importance for the success of a software project [8]. Nevertheless, there exists evidence that NFRs tend to come second class to functional requirements [8] [7]. We see this as a pervasive problem, regardless of the methodology that the development process follows. Quality management models and standards like ISO 9001:2015 [14] and CMMI [6] require that functional and non-functional requirements are documented as way of conveying their importance. These software development models and standards take a “do as you say, say as you do” approach where documentation and upfront planning is used to mitigate the risk of not delivering the software product within the constraints of the project. In fact, the ISO/IEC/IEEE 29148:2018 standard for software and systems requirements engineering [16], prescribes that both functional and non-functional requirements have to be documented.

Agile software engineering highlights the need of “continuous attention to technical excellence” [2]. Agile software engineering methods mainly rely on immediate feedback and postulate the sufficient availability of knowledgeable software developers to mitigate potential quality risks. Unfortunately, agile values and principles often seem to be adopted naïvely [11], i.e., equating agile with avoiding documentation [25].

The starting point of the research presented in this paper is the assumption that in order to be able to adequately handle NFRs, they must be documented –otherwise developers would not know about their precise nature or even their existence. Based on a subset of data acquired from the most recent NaPiRE (Naming the Pain in Requirements Engineering) survey conducted in 2018 [19], we calculate for the NFR types Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, and Usability how often respondents state they document a specific type of NFR when they also state that this type of NFR is important. In addition, we calculate the occurrence of potential Technical Debt (i.e., NFR not documented although labeled as important) and the occurrence of potential Waste (i.e., NFR documented although labeled as not important).

Our results show a clear indication of Technical Debt in several NFRs, with Maintainability, Reliability, Usability, and Performance being the NFRs with the highest frequency of occurrence of potential Technical Debt. Furthermore, when breaking down the analysis by the type of development process, the development processes at the extremes of the spectrum (i.e., purely plan-driven or purely agile) alter the indication of the Technical Debt pattern. Furthermore, our results show that there is less risk of incurring in Waste. Less than 15% of the respondents stated that they document NFR which they do not consider important. With Security being the NFR with the highest frequency of respondents at about 15%.

The rest of this paper is structured as follows. Section 2 provides the relevant background. Section 3 presents the applied research method and Section 4 shows the results. Section 5 discusses the results and threats to validity. Finally, Section 6 concludes the paper.

2 Background

In this section, we introduce the **N**aming the **P**ain in **R**equirements **E**ngineering (NaPiRE) initiative⁵ and present an overview on research about NFRs.

2.1 The NaPiRE Project

The objective of the NaPiRE project is to establish a comprehensive theory of requirements engineering (RE) practice and to provide empirical evidence to practitioners that helps them address the challenges of requirements engineering in their projects. These objectives shall be achieved by collecting empirical data in surveys conducted world-wide and in repeating cycles. At the time of writing this paper, three rounds of the NaPiRE survey have been carried out. The first survey round was conducted in Germany and the Netherlands in 2012 [21]. The second round, conducted in the years 2014 and 2015, was extended to ten countries [20]. The third round of the survey was conducted in 2018 and collected data from 42 countries. The research presented in this paper is based on the data collected in the third round. Since the NaPiRE survey instrument has evolved since 2014/2015, a direct comparison between past analysis results and currently ongoing analyses is not always possible. This holds especially for the topic of non-functional requirements covered in this paper, which is the first one published on data from the third run.

The previous installments of the NaPiRE survey have been successful in sparking complementing research into several viewpoint of requirements engineering. For instance, to compare requirements engineering practices across geographical regions [22][17] or by development method [28][29]. We argue that, although NaPiRE data has been extensively analyzed, it has so far not been analyzed with regards to practitioners' perceptions about handling NFRs.

2.2 Published Research on Non-Functional Requirements

This section presents an overview of past directions in non-functional requirements (NFR) research with a focus on survey research in the context of software industry.

Borg et al. [5] presented a case study on how NFRs are dealt with in two software development organisations. The authors interviewed 14 software developers in two organisations. Their results show that, in both contexts, functional requirements take precedence over non-functional requirements.

Berntsson Svensson et al. [4] investigated the challenges for managing NFRs in embedded systems. They interviewed ten practitioners from five software companies. Their results show a widespread variation in how the respondents dealt with NFR, they also suggest a relationship between a lack of documentation of NFR and a dismissal of NFR during the project lifecycle. Behutiye et al. [3] investigated how software development teams using agile projects deal with

⁵ NaPiRE web site – <http://napire.org>

non-functional requirements. The authors interviewed practitioners in four companies developing software with agile methodologies. Each company followed a different practices when documenting NFR (including not documenting them and relying on tacit knowledge).

Ameller et al. [1] looked at how software architects deal with non-functional requirements. Their results highlight a lack of common vocabulary among software architects to convey NFR, the two most important NFR types were performance and usability, and that NFRs are often not documented, and when documented, the documentation was usually imprecise and was rarely maintained. Also, Proot et al. [23] presented a survey about the perceived importance of non-functional requirements among software architects. Their results suggest that architects consider NFRs important to the success of their software projects.

De la Vara et al. [26] present a questionnaire-based survey capturing the more important NFRs from the point of view of practitioners. 31 practitioners from 25 organizations were selected within the industrial collaboration network of the authors. The top five NFRs identified are Usability, Maintainability, Performance, Reliability, and Flexibility. Haigh et al. [10] empirically examined the requirements for software quality held by different groups involved in the development process. She conducted a survey of more than 300 current and recently graduated students of one of the leading Executive MBA programs in the United States, asking them to rate the importance of each of 13 widely-cited attributes related to software quality. The results showed the following ranking of NFRs: Accuracy, Correctness, Robustness, Usability, Integrity, Maintainability, Interoperability, Augmentability, Efficiency, Testability, Flexibility, Portability, Reusability.

In summary, we conclude that the topic of NFRs has been extensively researched but there is few evidence of how the NFRs are documented. Furthermore, the specific relationship between importance level and degree of documentation has not yet been investigated.

3 Research Method

Before describing the research method we first present our understanding of relevant concepts and assumptions about our research. Secondly, we introduce the terminology used in this paper. Then, we present our research questions. Finally, we describe the data extraction and analysis procedure that we followed in order to answer those research questions.

3.1 Concepts and Assumptions

According to the software product quality standard ISO 25010:2011 [15], a non-functional requirement is a “requirement that specifies criteria that can be used to judge the operation of a software system” [15]. The same standard defines a quality model for the evaluation of product quality of a software system. Within

this quality model, the quality attributes (also known as quality characteristics) are defined. A quality attribute is a specification of the stakeholders' needs (Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability). We argue that both terms, NFR and quality attribute, are related and often used interchangeably in industry, even though this is not correct according to the precise definitions of these terms. Upon careful consideration, in particular looking at how the NaPiRE survey instrument framed the questions related to NFRs, in this work we interpret NFRs to be all requirements that do not specify a functional behaviour. Furthermore, we do not differentiate between NFR and quality attribute. We claim that (1) the NaPiRE questionnaire has not made this distinction evident, (2) most practitioners would not care for the subtleties of this differentiation, and (3) interchangeable use of terms is pervasive among practitioners and researchers [3], [23], [7]. In order to be consistent with the survey instrument used in the NaPiRE survey, in this paper, we use the term "quality attribute" instead of "NFR" when we present our research questions and the results of our analyses.

This research is driven by our assumption that, in agreement with [16], both functional and non-functional requirements have to be documented. In the context of software quality assurance, which is defined in ANSI/IEEE Standard 729-1983 [12], the confidence of the established technical requirements is achieved by checking the software and the documentation and verifying their consistency.

Therefore, the ideal situation is that when a quality attribute is considered as important for the development project, then it must be documented. To better convey this understanding we refer to the Technical Debt metaphor. Technical Debt, as defined by [24], is "a metaphor for immature, incomplete, or inadequate artefacts in the software development lifecycle that cause higher costs and lower quality in the long run. These artefacts remaining in a system affect subsequent development and maintenance activities, and so can be seen as a type of debt that the system developers owe the system." Also, Zengyang et al. [18] pointed out that documentation of Technical Debt refers to insufficient, incomplete, or outdated documentation in any aspect of software development. That is, when practitioners perceive a quality attribute as important but fail to document requirements associated to the quality attribute, we will interpret this as an indication of the incurred Technical Debt. We differentiate from [9], which defines Technical Debt in requirements as the distance between the implementation and the actual state of the world.

We follow similar reasoning on the other end of the spectrum but we rely on the concept of Waste in Lean development. In Lean development, Waste is defined as anything that does not add value [13]. In the domain of software development, the types of Waste can be interpreted as: extra features, waiting, task switching, extra processes, partially done work, movement, defects, or unused employee creativity [30]. Therefore, when practitioners are investing effort in documenting requirements for artifacts (quality attributes) that they do not consider important, we are interpreting that such an effort could be better placed elsewhere in the development process, and understand it as a source of Waste.

In the most recent round of the NaPiRE survey, practitioners were asked about their perception of importance regarding a set of pre-defined quality attributes in the context of the project they were currently working on. In addition, they were asked whether they document quality attributes. The specific questions related to these aspects and their possible responses are shown in Table 1. Question Q1 asks for the level of importance of each NFR type and Q2 for its degree of documentation. Questions Q3, Q4, and Q5 request the context factors project size, system type, and development process type, respectively. By combining the answers to Q1 and Q2, we can investigate if practitioners are following the requirements documentation recommendation for quality attributes in a specific context determined by Q3, Q4, and Q5.

Table 1. NaPiRE Questionnaire items used for the analysis

ID	Questionnaire item	Possible responses	Variables
Q1	Are there quality attributes which are of particularly high importance for your development project? If yes, which one(s)?	Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, Usability	v_6-v_13
Q2	Which classes of non-functional requirements do you explicitly consider in your requirements documentation?	Compatibility, Maintainability, Performance, Portability, Reliability, Safety, Security, Usability	v_97-v_102, v_303, v_103
Q3	How many people are involved in your project?	Free text	v_3
Q4	Please select the class of systems or services you work on in the context of your project.	Software-intensive embedded systems, Business information systems, Hybrid of both software-intensive embedded systems and business information systems	v_4
Q5	How would you personally characterize your way of working in your project?	Agile, Rather agile, Hybrid, Rather plan-driven, Plan-driven	v_24

Table 2 conveys our perception of the possible scenarios. In the ideal world, practitioners do not incur in Technical Debt (Important and Not Documented), nor do they Waste effort in documenting requirements which they do not consider important (Not Important and Not Documented (NI_ND)). However, our experience leads us to expect that, practitioners are restricted by the context of their development projects and they are bound to incur in Technical Debt and Waste. In this research, we will look for evidence of this understanding in the responses to the NaPiRE 2018 survey.

Table 2. Perception of importance and availability of documentation quadrant for analysis

Perception of importance	Documentation Available	
	Important and Documented (I.D) Expected situation	Important and Not Documented (I.ND) An Indication of Technical Debt
	Not Important and Documented (NI.D) An Indication of Waste	Not Important and Not Documented (NI.ND) Expected Situation

3.2 Research Questions

As mentioned in section 3.1, we argue that if a quality attribute is perceived important, then it should be documented. We have, therefore, divided our analysis into the following research questions:

RQ1: *Can we identify Technical Debt and Waste in requirements documentation (as interpreted in section 3.1) from the responses in the NaPiRE questionnaire?* This question expresses our overarching objective of understanding the juxtaposition between the perception of the importance of a quality attribute and if it has been documented. The question is framed in the Technical Debt metaphor, as it conveys our understanding that: “If a quality attribute is considered important, then it should be documented”. Any deviation in this direction should be interpreted as a project decision that, for whatever reason, lead the practitioners into not documenting a quality attribute they consider important (i.e., an expression of Technical Debt). Likewise, “if a quality attribute is not considered important, then it need not be documented”. Any deviation in this direction we consider as an indication of Waste, as the effort invested in documenting the quality attribute, could have been better spent elsewhere in the development lifecycle. **RQ1** is divided into:

RQ1.1: *For which quality attributes do the practitioners’ responses indicate Technical Debt?* Through this sub-question, we will explore practitioners’ responses to the NaPiRE 2018 dataset and identify the quality attributes in which a deviation is present of a quality attribute is perceived important and yet, it is not documented (referred in the analysis as *I.ND*).

RQ1.2: *For which quality attributes do the practitioners’ responses indicate Waste?* Through this sub-question, we will explore the practitioners responses to the NaPiRE 2018 dataset and identify the quality attributes in which a deviation is present of a quality attribute that is not perceived as important and yet, it has been documented (referred in the analysis as *NI.D*).

RQ2: *How does the practitioners’ context influence the occurrence of Technical Debt and Waste in requirements documentation?* This second research question conveys our pre-conception that practitioners fail to document some quality attributes that they consider important. **RQ2** is divided into:

RQ2.1: *How does the system type influence the occurrence of Technical Debt and Waste?* This question conveys our pre-conception that the type of system

can have an influence on the perceived importance of a quality attribute, and therefore on the occurrence of Technical Debt or Waste.

RQ2.2: *How does the project size influence the occurrence of Technical Debt and Waste?* This question conveys our pre-conception that the size of the software project can have an influence on the perception of importance or the documentation needs of a quality attribute.

RQ2.3: *How does the type of development process influence the occurrence of Technical Debt and Waste?* This question conveys our pre-conception that the development process type might have an influence on the perception of importance or the documentation needs of a quality attribute.

3.3 Data Extraction and Analysis Procedure

We base our analysis on the NaPiRE 2018 dataset and, thus, have access to the corresponding raw data as well as the pre-processed codification of the questionnaire and answers. Table 1 presents the variables included in this research.

A total of 488 responses are recorded for the NaPiRE 2018 instance of the survey. All recorded responses are complete for variables v_6 to v_13 (perceived importance of quality attributes, see Table 1) whereas only 455 responses are complete for variables v_97 to v_102, v_303, v_103 (documentation of requirements for quality attributes, see Table 1). We decided to remove other 57 responses for incompleteness in other variables of interest. Therefore, the total number of responses considered for this research is 398. Table 3 presents the distribution of responses in the aforementioned categories by the type of quality attribute.

Table 3. Distribution of responses by quality attribute

Quality attribute	Technical Debt (I_ND)	Waste (NI_ID)	I_ID	NI_ID
Compatibility	70 (17.59%)	46 (11.56%)	99 (24.87%)	183 (45.98%)
Maintainability	123 (30.9%)	27 (6.78%)	105 (26.38%)	143 (35.93%)
Performance	90 (22.61%)	47 (11.81%)	143 (35.93%)	118 (29.65%)
Portability	46 (11.56%)	39 (9.8%)	31 (7.79%)	282 (70.85%)
Reliability	122 (30.65%)	31 (7.79%)	117 (29.4%)	128 (32.16%)
Safety	68 (17.09%)	31 (7.79%)	39 (9.8%)	260 (65.33%)
Security	80 (20.1%)	59 (14.82%)	125 (31.41%)	134 (33.67%)
Usability	97 (24.37%)	35 (8.79%)	158 (39.7%)	108 (27.14%)
Mean	87.0 (21.86%)	39.375 (9.89%)	102.125 (25.66%)	169.5 (42.59%)

The distribution for the contextual project information that will be analyzed for **RQ2** is shown in Table 4. It is worth mentioning that we applied a pre-processing step to variable v_3 since it represents a free-text response. We used the results from the variable coding made by the collaborators of the NaPiRE

initiative during their data analysis phase. For the purpose of analysing this variable, we decided to group the responses into equal-sized buckets that represent small-sized ($v_3 < 7$), medium-sized ($7 \leq v_3 < 15$) and large-sized projects ($v_3 \geq 15$).

Table 4. Distribution of responses with regard to the variables under study. Mean and standard deviation are reported for the percentages of responses indicating Technical Debt and Waste.

Variable	Value	Technical Debt (I_ND)	Waste (NI_D)	Responses
Process type	Agile	26.59 ± 0.09	7.54 ± 0.03	63
	Rather agile	23.16 ± 0.10	10.26 ± 0.05	95
	Hybrid	20.28 ± 0.06	9.81 ± 0.03	135
	Rather plan-driven	18.57 ± 0.06	12.32 ± 0.04	68
	Plan-driven	22.30 ± 0.10	8.78 ± 0.05	37
Project size	S	24.01 ± 0.08	8.27 ± 0.02	139
	M	20.50 ± 0.07	11.49 ± 0.04	136
	L	20.93 ± 0.06	9.96 ± 0.03	123
System class	BIS	22.40 ± 0.09	9.65 ± 0.04	202
	HYB	23.14 ± 0.05	8.91 ± 0.03	101
	SIES	19.34 ± 0.05	11.45 ± 0.02	95

4 Results

This section presents the results of our analysis organized by the research questions.

RQ1: *Can we identify Technical Debt and Waste (as interpreted in Section 3.1) from the responses in the NaPiRE questionnaire?* To answer RQ1 we cross-reference the responses to the perceived importance of quality attributes (v_6 to v_13 , see Table 1) with the availability of documentation (v_97 to v_102 , v_303 , and v_103 , see Table 1). Important and not documented (I_ND) requirements indicate Technical Debt, whereas not important and documented (NI_D) requirements indicate Waste (see Table 2).

RQ1.1: *For which quality attributes do the practitioners' responses indicate Technical Debt?* Table 3 shows the occurrence of Technical Debt for each quality attribute. The percentage of responses showing Technical Debt (I_ND) ranges from 12% to 31%. The average percentage of responses related to Technical Debt over all quality attribute types is 22%. The quality attributes which are most likely to incur in Technical Debt are Reliability (31%), Maintainability (31%), Usability (24%), and Performance (23%).

RQ1.2: *For which quality attributes do the practitioners' responses indicate Waste?* Table 3 shows that waste also occurs in all quality attributes (albeit

at a smaller response rate). The percentage of responses showing Waste (N.I.D) ranges from 7% to 15%. The quality attributes which exhibit higher Waste are Security (15%), Performance (12%), and Compatibility (12%).

RQ2: *How does the practitioners context influence the occurrence of Technical Debt and Waste?* To answer RQ2, we blocked the response data by the variables type of system, project size, and development process type to investigate their influence on the occurrence of Technical Debt and Waste. Figure 1 shows the percentage of responses by quality attribute that indicate Technical Debt for each of the variables under study. Similarly, Figure 2 shows the percentage of responses related to Waste.

RQ2.1: *How does the system type influence the occurrence of Technical Debt and Waste?* When broken down by the system type (see Figure 1 (a)) we can observe that Reliability is the most prone to Technical Debt in all types of systems. On the other end, Portability, is not prone to Technical Debt in the system types under analysis. The HYB type of system seems to be the type of system where the average percentage of responses indicating Technical Debt is the highest (23%) (see Table 4). Four quality attributes surpass this value, namely Usability (25%), Reliability (31%), Performance (27%), and Maintainability (29%). Security (23%) can be considered a borderline case. The percentage of BIS showing Technical Debt ranges from 9% to 37% with highest values for Maintainability (37%), Reliability (32%), and Usability (26%). From Figure 1 (a) we can see that three BIS systems surpass the average percentage of responses for all the quality attributes (22%), namely Usability (26%), Reliability (32%), and Maintainability (37%). This system type also shows the highest percentages for Maintainability (37%) and Reliability (32%). Finally, SIES systems show percentages of Technical Debt ranging from 12% to 28%, and four quality attributes surpass the average percentage of responses (19%), namely, Reliability (28%), Performance (22%), Security (21%), and Usability (21%). Maintainability (20%) can be considered as a borderline case.

Regarding Waste, the average percentage of responses for all the quality attributes is 10%, 9%, and 11% for BIS, HYB, and SIES type of systems (see Table 4). When broken down by the system type (see left-side of Figure 2 (a)), the highest percentages of responses are related to the Security of BISs (18%) and the Compatibility of SIES (16%). At the other end, the lowest percentage is related to the Maintainability of HYB systems (4%).

RQ2.2: *How does project size influence the occurrence of Technical Debt and Waste?* Figure 1 (b) shows the Technical Debt for each quality attribute blocked by project size (S, M, L). Similarly, Figure 2 (b) shows the Waste. The average percentages of responses indicating Technical Debt is 24%, 20%, and 21% for projects of size S, M, and L, respectively (see Table 4).

Maintainability, and Reliability are the quality attributes which show the highest percentages of Technical Debt (regardless of project size). On the other end, Portability is the quality attribute with the lowest Technical Debt regardless of project size. Small projects incur in Technical Debt having percentages ranging from 13% to 35%. This kind of projects particularly shows high percentages

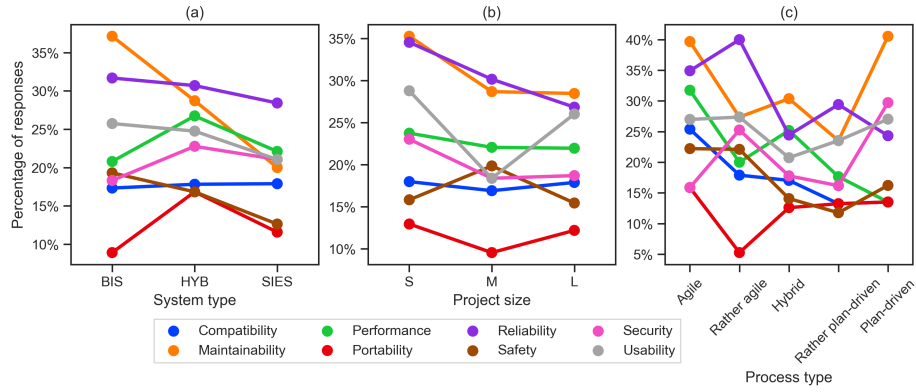


Figure 1. Percentage of responses indicating Technical Debt by System type, Project size, and Process type.

related to Reliability (35%) and Maintainability (35%). As for medium-sized projects, the percentages range from 10% to 30%. In large-sized projects, the percentages of Technical Debt range from 12% to 28%.

Regarding Waste, the average percentages of responses indicating Waste is 8%, 11%, and 10% for projects of size S, M, and L, respectively (see Table 4). The percentages range from 5% to 12% for small-sized projects, from 7% to 17% for medium-sized projects, and from 7% to 16% for large-sized projects. In particular, the data points for Security and Safety seem to indicate that the number of responses showing Waste becomes larger as the project size increases.

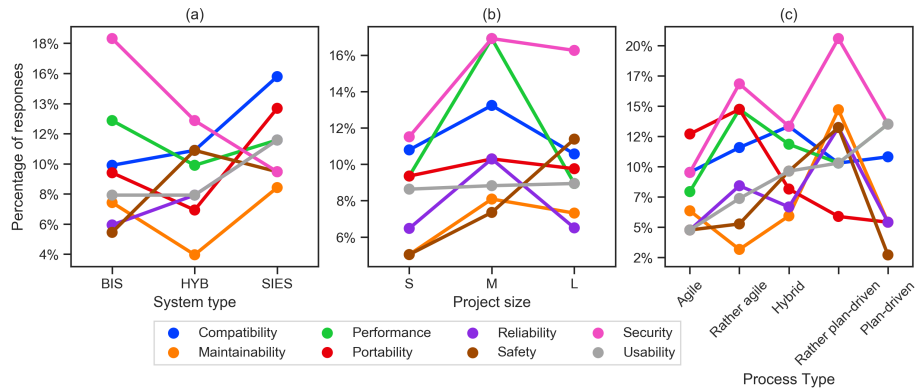


Figure 2. Percentage of responses indicating Waste by System type, Project size, and Process type.

RQ2.3: *How does type of development process influence the occurrence of Technical Debt and Waste?* Figure 1 (c) shows the percentages of Technical Debt for every quality attribute organized by development process types. Similarly, Figure 2 (c) shows the percentages related to Waste. Three quality attributes exhibit the highest percentages of Technical Debt regardless of the type of development process, namely Maintainability, Reliability, and Usability. On the other hand, Portability is the only quality attribute without Technical Debt for any type of development process.

Regarding Waste, the percentage responses indicating Waste related to Security is the highest. In addition, the percentage seems to increase as the projects become more plan-driven. The development process characterised as *Rather plan-driven* shows the highest overall exposure to Waste.

5 Discussion

In this section we first discuss the results achieved regarding the relation between NFRs with Technical Debt and Waste, respectively. In addition, we discuss possible threats to validity of our study.

Observations Related to NFR and Technical Debt Our results show that the majority of the participants of the survey stated that they document NFRs when they are important and they don't document NFRs when they are not important. This is what we had hypothesized. However, we observed that there is a substantial subset of respondents who stated that they don't document (some of the) important NFRs. This is what we interpret as being at risk of Technical Debt. Certain types of NFRs were particularly prone to this phenomenon, i.e., Reliability, Maintainability, Usability, and Performance. We can only speculate what would drive practitioners into this behaviour. For example, either these NFR types are difficult to document, knowledge on how to properly document NFRs is missing, or no appropriate tool is available. Furthermore, reasons might vary by NFR type. For example with *Maintainability*, it can be argued that is left to good coding practices (i.e., avoiding code smells and focusing on refactoring). This phenomenon might also be true for other NFR types, i.e., there exist standard procedures or standard requirements that always hold and do not have to be explicitly stated in each individual project. When respondents answered the NaPiRE questionnaire, they might only have thought about project-specific documentation of NFRs.

Observations Related to NFRs and Waste We also observed the occurrence of Waste, i.e., cases where respondents stated they document NFRs although they are not considered important. However, the observed Waste was consistently smaller than the Technical Debt (for the same quality attribute). Furthermore, when looking at the percentages observed for Waste, the proportion of Waste increases as project size increases: 5-11% for small-sized projects, 7-17% for medium-sized projects, and 7-16% for large-sized projects. This might be a signal that - consistent with common expectation - for larger projects the

risk of Waste is higher than for small projects with respect to NFRs. Surprisingly, and probably against common expectation, our analyses do not give any indication that projects using rather agile or purely agile processes produce less Waste than projects using plan-driven development approaches.

Threats to Validity We consider threats to construct, internal, external and conclusion validity according to Wohlin et al. [31] as well as measures to mitigate them.

This research is grounded on the NaPiRE 2018 survey, therefore we inherit some of the decisions taken during the development of the survey instrument. Of particular importance to the research presented in this paper is the fact that the NaPiRE 2018 survey does not differentiate between quality attribute and NFR. Both concepts are confounded in the questions on which we based our analysis. As a research team we have discussed this issue in depth and decided to accept this threat as it is in line with our shared understanding that (1) we cannot revert this decision; (2) we share the understanding that practitioners would probably not differentiate between both (and even for those who do, we can probably not guarantee a shared understanding). The latter argument is in line with the results of Eckardt et al. [8] (already mentioned in Section 3.1) that there is a large variety in the understanding of what is quality and what are NFRs. Continuing with inherited threats, external validity of our results highly depends on the profile of participants in the NaPiRE survey. The survey received overall 488 responses from all over the world and we have shown in a previous paper [27] that there are no significant differences in the NaPiRE data with respect to different cultural regions. Furthermore, we analyzed the data also with respect to the system type, the project size and the type of development process. We therefore think that threats to external validity are low.

An important construct validity injected by the approach described in this work relates to how the metaphor of Technical Debt and the concept of Waste were introduced into the analysis of the data set. First of all, the NaPiRE survey makes no reference to these concepts. Secondly, there is a subtle but present gap between the formulation of the questions and our interpretation. It can be argued that "which quality attribute is of particular high importance?" (as asked in the survey) is not the same as asking "List all quality attributes that are important".

Regarding internal validity, a limitation that we always have with survey research is that surveys can only reveal perceptions of the respondents that might not fully represent reality. However, the analysis stems from the well-validated NaPiRE questionnaire (see Section 2.1), which has continuously been improved based on piloting and the first two runs. Furthermore, we tried to be explicit in our decision about our data cleaning criteria (see Section 3.3) to be able to perform a thorough analysis.

6 Conclusion

This paper explored the relationship between the level of importance and the degree of documentation for the NFR types Compatibility, Maintainability, Per-

formance, Portability, Reliability, Safety, Security, and Usability. The analysis is based on the data collected during the most recent run of the NAPiRE survey. To analyze this relationship we refer to the Technical Debt and Waste metaphors. To the best of our knowledge, this is the first publication in which these two concepts were explored in the context of NFRs. The starting point of our analysis was the assumption that important NFRs must be documented. If a project breaks this rule, then we interpret it as a possible source of Technical Debt. Likewise, we postulated that not important NFRs should not be documented. If a project breaks this rule, then we interpret it as a possible source of Waste. Our analyses indicate that for four types of NFR (Maintainability, Reliability, Usability, and Performance) more than 22% of the survey respondents who labelled the respective NFR type as important said that they did not document it. We interpret this as an indication that these NFR types have a higher risk of Technical Debt than other NFR types. Our analysis also indicates that the risk of Waste is less evident than the Risk of Technical Debt. Regarding Waste, NFR relating to Security exhibit the highest (about 15%) number of respondents that say that they do not consider Security important, but do document related requirements. For the remaining NFR under analysis, the respondents indicate that the problem of Waste is much less evident (when compared to Technical Debt). Additional analyses indicate that our results are neither sensitive to the type of system class nor to project size. However, the type of development process seems to influence the results: extreme process types (i.e., purely plan-driven or purely agile) slightly alter the result patterns.

Overall, we conclude that, for specific NFR types (i.e., Maintainability, Reliability, Usability, and Performance), there is a clear indication that lack of documentation of important NFRs occurs regularly, pointing to the risk of Technical Debt. Regarding Waste, with the exception of Security, we conclude that the manifestation of Waste is not as clear as the manifestation of Technical Debt. We discussed several potential reasons for the occurrence of this phenomenon. However, investigating the true causes of Technical Debt for certain types of NFRs requires more empirical research, which we consider as future work.

Acknowledgments

The authors would like to thank all practitioners who took the time to respond to the NaPiRE survey as well as all colleagues involved in the NaPiRE project. The authors further acknowledge Dietmar Pfahl's contribution to research process described in this paper. Ezequiel Scott is supported by the Estonian Center of Excellence in ICT research (EXCITE), ERF project TK148 IT Tippkeskus EXCITE. Gabriela Robiolo is supported by Universidad Austral.

References

1. Ameller, D., Ayala, C., Cabot, J., Franch, X.: In: 2012 20th IEEE International Requirements Engineering Conference, RE 2012 - Proceedings. Chicago, USA

2. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al.: Manifesto for agile software development (2001)
3. Behutiye, W., Karhapää, P., Costal, D., Oivo, M., Franch, X.: In: Product-Focused Software Process Improvement. PROFES 2017. Lecture Notes in Computer Science. Innsbruck
4. Berntsson Svensson, R., Gorschek, T., Regnell, B.: In: REFSQ 2009: Requirements Engineering: Foundation for Software Quality
5. Borg, A., Yong, A., Carlshamre, P., Sandahl, K.: The Bad Conscience of Requirements Engineering : An Investigation in Real-World Treatment of Non-Functional Requirements. Third Conference on Software Engineering Research and Practice in Sweden (SERPS'03), Lund (2003)
6. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI : Guidelines for Process Integration and Product Improvement. Addison-Wesley, Upper Saddle River, NJ, second edn. (2007)
7. Chung, L., do Prado Leite, J.C.S.: On Non-Functional Requirements in Software Engineering, pp. 363–379. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
8. Eckhardt, J., Vogelsang, A., Fernández, D.M.: Are "non-functional" requirements really non-functional? In: Proceedings of the 38th International Conference on Software Engineering - ICSE '16. pp. 832–842. ACM Press, New York, New York, USA
9. Ernst, N.A.: On the role of requirements in understanding and managing technical debt. In: Proceedings of the Third International Workshop on Managing Technical Debt. pp. 61–64. Piscataway, NJ, USA (2012)
10. Haigh, M.: Software quality, non-functional software requirements and it-business alignment. *Software Quality Journal* **18**, 361–385 (2010)
11. Hoda, R., Noble, J.: In: Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017
12. IEEE: IEEE standard glossary of software engineering terminology. IEEE Std 610.12-1990 pp. 1–84 (Dec 1990)
13. Ikonen, M., Kettunen, P., Oza, N., Abrahamsson, P.: Exploring the sources of waste in kanban software development projects. In: 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications. pp. 376–381 (2010)
14. ISO: ISO 9001:2015. Quality Management Systems - Requirements (2015)
15. ISO/IEC Standard: ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011)
16. ISO/IEC/IEEE: ISO/IEC/IEEE 29148:2018 Systems and software engineering – Life cycle processes – Requirements engineering. Tech. rep., International Standards Organization (2018)
17. Kalinowski, M., Felderer, M., Conte, T., Spínola, R., Prikladnicki, R., Winkler, D., Fernández, D.M., Wagner, S.: Preventing incomplete/hidden requirements: reflections on survey data from Austria and Brazil. In: International Conference on Software Quality. pp. 63–78. Springer (2016)
18. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. *Journal of Systems and Software* **101**, 193–220 (2015)
19. Méndez Fernández, D.: Supporting Requirements-Engineering Research That Industry Needs: The NaPiRE Initiative. *IEEE Software* **35**(1), 112–116 (jan 2018)
20. Méndez Fernández, D., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.T., Greer, D., Lassenius, C., Männistö, T., Nayabi,

- M., Oivo, M., Penzenstadler, B., Pfahl, D., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spinola, R., Tuzcu, A., de la Vara, J.L., Wieringa, R.: Naming the pain in requirements engineering. *Empirical Software Engineering* **22**(5), 2298–2338 (oct 2017)
21. Méndez Fernández, D., Wagner, S.: Naming the pain in requirements engineering: design of a global family of surveys and first results from Germany. In: *EASE - International Conference on Evaluation and Assessment in Software Engineering* (17th, 2013, Porto de Galinhas) (2013)
22. Méndez Fernández, D., Wagner, S., Kalinowski, M., Schekelmann, A., Tuzcu, A., Conte, T., Spinola, R., Prikladnicki, R.: Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany. *IEEE Software* **32**(5), 16–23 (sep 2015)
23. Poort, E.R., Martens, N., van de Weerd, I., van Vliet, H.: How Architects See Non-Functional Requirements: Beware of Modifiability. In: Regnell, B., Damian, D. (eds.) *Requirements Engineering: Foundation for Software Quality*. pp. 37–51. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
24. Seaman, C., Guo, Y.: Chapter 2 - measuring and monitoring technical debt. *Advances in Computers* **82**, 25–46 (2011)
25. Stettina, C.J., Heijstek, W.: Necessary and neglected? An empirical study of internal documentation in agile software development teams. In: *Proceedings of the 29th ACM International Conference on Design of Communication (SIGDOC 2011)*, Pisa, Italy 3-5 October, 2011 (2011)
26. de la Vara, J.L., Wnuk, K., Berntsson Svensson, R., Sanchez, J., Regnell, B.: An empirical study on the importance of quality requirements in industry. In: *Proceedings of 23rd International Conference on Software Engineering and Knowledge Engineering*. pp. 311–317. New York, NY, USA (2010)
27. Wagner, S., Fernández, D.M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.T., Greer, D., et al.: Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)* **28**(2), 9:1–9:48 (2019)
28. Wagner, S., Méndez Fernández, D., Felderer, M., Kalinowski, M.: Requirements Engineering Practice and Problems in Agile Projects; Results from an international survey. In: *2017 Iberoamerican Conference on Software Engineering (CiBSE 2017)* (2017)
29. Wagner, S., Méndez-Fernández, D., Kalinowski, M., Felderer, M.: Agile requirements engineering in practice: Status quo and critical problems. *CLEI Electronic Journal* **21**(1) (2018)
30. Wang, X., Conboy, K., Cawley, O.: leagile software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software* **85**, 1287–1299 (2012)
31. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: *Experimentation in software engineering*. Springer Science & Business Media (2012)